DB-BERT: A Transformer-Based Approach for Predicting SQL Query Difficulty from Natural Language Questions

Debayan Dutta

Vellore Institute of Technology, Vellore

October 30, 2025

Abstract

Natural Language Interfaces (NLIs) to databases enable non-technical users to interact with complex data systems through natural language queries. However, the lack of mechanisms to proactively assess query complexity poses significant operational challenges. A seemingly simple user question can translate into a highly expensive SQL query, straining database resources and degrading system performance. This work proposes DB-BERT, a robust Transformer-based model for predicting SQL query difficulty directly from natural language, *prior to query generation*. Our approach frames difficulty detection as a four-class classification problem, leveraging label rules grounded in syntactic SQL features using the Spider dataset. DB-BERT, fine-tuned from BERT, achieves **81.02% Macro F1-score**, outperforming a strong TF-IDF + SVM baseline by **6.54 percentage points**. These findings demonstrate that deep semantic models can effectively learn query complexity patterns from user utterances, enabling NLIs to anticipate resource needs, route costly queries appropriately, and enhance overall system throughput and reliability.

Keywords: Natural language processing, text-to-SQL, query performance prediction, BERT, transformers, database management systems, query optimization.

1 Introduction

The proliferation of Natural Language Interfaces (NLIs) to databases has democratized data access, enabling non-technical users to query complex systems without SQL expertise. While recent advances in sequence-to-sequence models have significantly improved SQL generation accuracy, a critical operational challenge remains: database systems lack foresight regarding a query's computational burden before execution. Simple requests (e.g., `SELECT name FROM users WHERE id = 1`) execute effortlessly, while complex analytical queries—involving multiple joins, aggregations, or nested subqueries—can monopolize compute resources, creating bottlenecks and degrading service quality across the entire system.

Consider the operational contrast between two semantically distinct user questions:

(a) *"List all employees in department X"*

(b) *"For each department, find the monthly average salary, sorted by departments with the highest variance"*

The first query incurs minimal cost, while the second requires aggregations, sorting, and potentially complex nested operations. In production environments, treating both uniformly risks resource exhaustion and impacts all concurrent users. An NLI capable of distinguishing query complexity *before* SQL generation or execution can enable intelligent, real-time resource management decisions.

1.1 Motivation

The research motivation is threefold:

Resource Management: Enable DBMS systems to anticipate query cost *a priori* and allocate computational resources accordingly, preventing resource starvation for concurrent queries.

User Experience: Equip NLIs with contextual feedback mechanisms to inform users about expected query latency (e.g., “This is a complex analytical query that may take several minutes”), managing expectations proactively.

System Stability: Prevent resource exhaustion by routing expensive analytical queries to read-replicas or OLAP systems, protecting the performance and availability of primary OLTP databases serving real-time applications.

1.2 Contributions

This work introduces DB-BERT, a Transformer-based classifier that predicts SQL query difficulty from natural language input alone. Our principal contributions include:

- A novel framing of query complexity prediction as a supervised multi-class classification problem operating on natural language, enabling pre-generation intervention at the earliest possible stage.
- A systematic, rule-based labeling scheme for the Spider dataset grounded in SQL syntactic features, creating a reproducible benchmark for this emerging task.
- Empirical validation demonstrating that DB-BERT achieves 81.02% Macro F1-score, significantly outperforming classical machine learning baselines and establishing the feasibility of linguistic complexity prediction.
- Analysis revealing that natural language structure alone contains sufficient semantic signal for accurate query complexity prediction without access to database schema or statistics.

2 Related Work

2.1 Text-to-SQL Generation

Pioneering datasets such as Spider [1] have catalyzed substantial advancements in semantic parsing by providing large-scale, cross-domain benchmarks with complex SQL queries. State-of-the-art models like RAT-SQL [2], which incorporates relation-aware self-attention over database schemas, and large-scale pre-trained transformers including T5-based approaches (e.g., PICARD [3]) and BERT [4], now achieve high exact-match and execution accuracy rates.

However, existing Text-to-SQL research fundamentally prioritizes generation *correctness*, systematically omitting considerations of query execution cost, resource consumption, or operational impact. These models aim to produce syntactically valid and semantically correct queries without addressing whether those queries are computationally tractable in production environments.

2.2 Query Performance Prediction

Within database systems research, Query Performance Prediction (QPP) has been extensively studied, but predominantly from the perspective of already-generated queries or execution plans. Traditional query optimizers employ cost-based models that estimate execution time post-generation, frequently yielding substantial errors due to optimizer limitations, stale statistics, complex data distributions, or correlated predicates.

More recent machine learning approaches attempt to build learned cost models [4], but these invariably require either the full SQL query text or the execution plan as input. Our work fundamentally differs by predicting difficulty directly from the natural language question *before* SQL generation, enabling intervention at the earliest possible stage and completely circumventing the need for query plan generation or analysis.

3 Methodology

We formalize query difficulty prediction as a supervised multi-class text classification problem: given a natural language user input Q_{NL}, predict a difficulty label $\hat{y} \in \{0, 1, 2, 3\}$ representing increasing levels of SQL complexity.

3.1 Dataset and Label Generation

The Spider 1.0 dataset [1] provides 10,181 paired (question, SQL) samples spanning 200 databases across diverse domains. While Spider lacks explicit difficulty annotations, we design a systematic rule-based labeling scheme (Table 1) mapping SQL syntactic features to four difficulty tiers. We combine `train_spider.json` and `train_others.json` for 8,659 training samples and use `dev.json` for 1,034 test samples.

Table 1: Difficulty Label Definitions

Label	**Name**	**SQL Features**
0	Easy	Simple `SELECT-FROM-WHERE`
1	Medium	Contains `JOIN`, `GROUP BY`, `ORDER BY`, or `LIMIT`
2	Hard	Exactly two `SELECT`s (subquery) or `HAVING`
3	Extra Hard	>2 `SELECT`s or set operations (`UNION`, `INTERSECT`, `EXCEPT`)

The resulting distribution (Table 2) exhibits significant class imbalance, with "Medium" comprising over half of all samples. This imbalance motivates our selection of Macro F1-score as the primary evaluation metric, ensuring equal treatment across all difficulty levels.

Table 2: Training Set Label Distribution (N=8,659)

Label	**Name**	**Count**	**%**
0	Easy	2,377	27.45
1	Medium	4,525	52.26
2	Hard	1,158	13.37
3	Extra Hard	599	6.92
Total		**8,659**	**100.00**

3.2 Baseline Model: TF-IDF + SVM

To establish a strong classical baseline, we implement a TF-IDF feature vectorization coupled with a Support Vector Machine classifier. Each question Q_{NL} is transformed into a sparse feature vector $\mathbf{x} \in \mathbb{R}^{5000}$ using up to 5,000 unigrams and bigrams weighted by term frequency-inverse document frequency. A linear SVM with L_2 regularization is trained to predict difficulty classes. This baseline effectively captures keyword-based patterns (e.g., "average of each" indicating aggregation operations).

3.3 DB-BERT Architecture

DB-BERT leverages a pre-trained BERT-base-uncased model [4] fine-tuned for four-class classification. The architecture comprises:

1. **Tokenization:** Natural language input Q_{NL} undergoes WordPiece tokenization:

$$X = \{[\text{CLS}], t_1, t_2, \ldots, t_n, [\text{SEP}]\} \tag{1}$$

2. **Contextualized Encoding:** A 12-layer bidirectional Transformer encoder f_{BERT} produces contextualized hidden states:

$$H = f_{\text{BERT}}(X) \in \mathbb{R}^{(n+2)\times d_h} \tag{2}$$

   where $d_h = 768$ represents the hidden dimension.

3. **Sequence Representation:** The final hidden state of the `[CLS]` token serves as the aggregate sequence representation:

$$\mathbf{v} = h_{[\text{CLS}]} \in \mathbb{R}^{768} \tag{3}$$

4. **Classification Head:** A randomly-initialized linear layer with softmax activation produces class logits:

$$\mathbf{z} = \mathbf{v}W_c + \mathbf{b}_c \tag{4}$$

   where $W_c \in \mathbb{R}^{768\times 4}$ and $\mathbf{b}_c \in \mathbb{R}^4$.

5. **Probability Distribution:** Softmax normalization yields class probabilities:

$$\hat{p}_j = P(\hat{y}{=}j \mid Q_{\text{NL}}) = \frac{\exp(z_j)}{\sum_{k=0}^{3}\exp(z_k)} \tag{5}$$

6. **Training Objective:** The model minimizes categorical cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{N}\sum_{j=0}^{3} y_{ij}\log(\hat{p}_{ij}) \tag{6}$$

   where y_{ij} is the one-hot encoded ground truth label.

4 Experiments and Results

4.1 Implementation Details

Data Splits: Training was performed on 8,659 samples (combined Spider `train_spider.json` and `train_others.json` splits). Evaluation was conducted on the 1,034 samples from the `dev.json` split.

Preprocessing: All questions were padded or truncated to a maximum sequence length of 128 tokens.

Hyperparameters: DB-BERT was fine-tuned using the AdamW optimizer with a learning rate of 2×10^{-5} and a batch size of 16. The model was trained for 4 full epochs, and the model state from the final epoch was used for evaluation.

Infrastructure: All experiments were conducted on a single NVIDIA T4 GPU via Google Colab. Total training time for the DB-BERT model was approximately 15 minutes.

4.2 Evaluation Metrics

We report two standard metrics:

- **Accuracy:** Overall percentage of correct predictions.
- **Macro F1-score** (primary): Unweighted average of per-class F1-scores.

The Macro F1-score is crucial for imbalanced datasets:

$$F1_j = \frac{2 \cdot P_j \cdot R_j}{P_j + R_j}, \quad F1_{\text{macro}} = \frac{1}{4}\sum_{j=0}^{3} F1_j \tag{7}$$

where P_j and R_j denote precision and recall for class j.

4.3 Main Results

Table 3 presents our main experimental findings. DB-BERT substantially outperforms the strong classical baseline, achieving 81.62% accuracy and 81.02% Macro F1-score. The +6.54 percentage point improvement in Macro F1 demonstrates that deep contextual understanding via Transformers provides tangible benefits over keyword-based pattern matching.

Table 3: Comparative Evaluation on Spider Dev Set ($N = 1{,}034$)

Model	**Acc.**	**Macro F1**	**ΔAcc**	**ΔF1**
TF-IDF + SVM	76.89%	74.48%	—	—
DB-BERT (Ours)	**81.62%**	**81.02%**	**+4.73%**	**+6.54%**

The substantial gap between DB-BERT and TF-IDF+SVM highlights the value of semantic understanding. Classical models treat “find all students” and “find the student with highest GPA” as similar (both contain “find” and “student”). In contrast, DB-BERT understands that “highest GPA” implies aggregation and sorting operations, correctly classifying it as a more complex query.

5 Discussion

Our experimental results validate the core hypothesis that natural language questions contain sufficient semantic and structural signals to accurately predict SQL query complexity. The 81.02% Macro F1-score represents strong performance given the nuanced differences between difficulty tiers, demonstrating that pre-trained Transformers can effectively learn linguistic patterns mapping to database operations.

5.1 Practical Implications

DB-BERT enables several high-impact production applications:

1. **Intelligent Query Routing:** Automatically direct predicted Easy queries to primary OLTP databases while routing Hard/Extra Hard queries to OLAP replicas or data warehouses, protecting real-time application performance.
2. **Proactive User Feedback:** Provide immediate contextual warnings ("This complex analytical query may take several minutes"), managing user expectations and improving perceived system responsiveness.
3. **Dynamic Resource Allocation:** Enable query schedulers to provision computational resources dynamically based on predicted complexity, optimizing cluster utilization.
4. **Educational Applications:** Integrate into SQL tutoring systems to help students understand why certain questions are inherently more complex to answer.

6 Conclusion and Future Work

This work presented DB-BERT, a Transformer-based classifier achieving 81.02% Macro F1-score for predicting SQL query difficulty from natural language. By validating that linguistic structure encodes query complexity signals, we demonstrate feasibility for proactive query management in production database systems. Our systematic labeling scheme and strong empirical results establish a foundation for intelligent resource-aware NLIs that can anticipate operational challenges before query execution, enabling more robust and scalable database access for non-technical users.

6.1 Future Directions

While this work establishes a strong baseline, several key limitations of our syntactic approach point to clear directions for future research:

- **Runtime Validation:** A primary next step is to correlate our syntactic labels with measured execution times across multiple database backends to validate their efficacy as a proxy for real-world cost.
- **Hybrid Models:** Future models could be enhanced by incorporating database statistics (e.g., table cardinalities, index availability) alongside the linguistic input for more dynamic and accurate cost estimation.
- **Live Deployment:** Integrating DB-BERT into a production NLI to measure real-world throughput improvements and system stability would provide a definitive test of its practical utility.
- **Fine-Grained Prediction:** The model could be extended to a regression task, predicting specific resource usage (e.g., CPU, memory, I/O) rather than discrete difficulty classes.
- **Open-Source Release:** We plan to publish our pre-trained DB-BERT models and the generated difficulty labels for the Spider dataset to foster community research and reproducibility.

Acknowledgment

The author thanks the creators of the Spider dataset for making their data publicly available and the NLP research community for developing and sharing pre-trained transformer models.

References

[1] T. Yu *et al.*, “Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Brussels, Belgium, 2018, pp. 3911–3921.

[2] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, “RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers,” in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Online, 2020, pp. 7567–7578.

[3] T. Scholak, N. Schucher, and D. Bahdanau, “PICARD: Parsing incrementally for constrained auto-regressive decoding from language models,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Punta Cana, Dominican Republic, 2021, pp. 9895–9901.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics (NAACL-HLT)*, Minneapolis, MN, USA, 2019, pp. 4171–4186.